

Fault Tolerant Operation in Aero Engine Using Distributed Computation System

Neela A G

M Tech Scholar,
VLSI Design & Embedded Systems,
VTU Regional Centre @ UTL
Technologies, Bangalore, India

Shobha S Prabhu

Scientist 'F', GTRE
DRDO, CV Raman Nagar,
Bangalore, India

Channabassappa Baligar

Scientist 'E', LRDE,
DRDO, C V Raman Nagar,
Bangalore, India

Abstract— The paper presents fault tolerant operation in an aero engine based on real-time systems which is built for a very small set of mission-critical applications like space craft's , avionics and other distributed control systems. The modern software deals with external interfaces and has to consider various timing implications. The platform is based on the C and developed using Keil MDK tool with the targeted deadline of 100 milliseconds at the baud rate of 500 kbps. CAN interface executes the role of Transportation and Communication, an interface cable used for serial communication between Digital Electronic Control Unit (DECU) and the host to transfer data to the pilot Online Monitoring System and that is based on Laboratory Virtual Instrument Engineering Workbench (Lab VIEW) 7.1. Fault diagnosis typically assumes a sufficiently large fault signature and enough time for a reliable decision to be reached. However, for a class of safety critical faults on commercial aircraft engines, prompt detection is paramount within a millisecond range to allow accommodation to avert undesired engine behavior. At the same time, false positives must be avoided to prevent inappropriate control action.

Keywords- Aero gas turbine Engine; Digital Electronic Control Unit (DECU); Online Monitoring display; CAN serial communication.

I. INTRODUCTION

Defence aircraft is a high-performance avionics mission computer which is the central computing resource. It evaluates, coordinates, controls large amount of data like rig, ground, altitude, temperature, fuel intake, power dissipated, pressure, amount of air taken, speed etc. It must also respond quickly to the changing conditions within the aircraft and in its own operating environment. An avionics mission computer has very strict "hard" Real Time requirements. Very low latency and exceedingly predictable behaviors are crucial not only to mission success, but also to the safety of the pilots. Any computer hardware or software designed for this environment, receives intense scrutiny for its performance and reliability [1].

In embedded system it's important that the behavior of the application has to be same on development environment (host PC) as well as deployed environment (target). Due to unavailability of target there is a need to check for application functionality, timing performance etc. The simulator development environment and operating system is different from target board where actual application is residing. In

embedded system executable application has to be deployed on the limited available size of the flash. Execution of application from FLASH creates unforeseen timing dependencies as well as other behaviors, which are tolerable in FLASH [1].

Presently, CAN interface cable is used for real-time communication between Gas turbine aero engine controller and Pilot critical display. Due to redundancy, there is a need for similitude component to provide alternative in case of any component failure. In order to achieve this, CAN serial communication is used in this application. Fault detection typically assumes a sufficiently large fault signature and enough time to come up with a reliable decision. However, faults that affect aircraft safety must be detected quickly. Otherwise, undesired consequences may arise such as engine surge stall events, power loss, severe vibrations, power loss, and no thrust response to commanded power. These events in addition to inappropriate crew response may lead to accidents [8]. Since the air traffic is projected to increase in the long term, and at the current level of reliability, the number of accidents would increase as well. Based on the review of an engine events database of the past 20 years of DEC and FADEC operated aircraft a list of root cause faults was compiled. From this list a set of representative cases from different categories including sensor faults, actuator faults was chosen. More specifically, compressor damage, turbine damage, variable stator vanes faults, and ps3 sensor loss were selected. The results of these failures can lead to compressor stall, loss of Power, loss of thrust control and flameouts. Requirements for time to detection were established to range from roughly 100ms for some actuator faults. This means that in the worst case, only a few samples of sensor data are available to perform the fault detection. With these time constraints, the full magnitude of the fault signature will in some cases not be seen in the measurement data. To allow for the detection of small changes in engine components, a suite of techniques was employed to address certain aspects of these issues in parallel.

The presented work is organized in six categories. Section II describes with Keil MDK and LPC 4357 features. It also gives a brief overview of Lab VIEW and serial communication. Section III deals with Aero gas turbine engine controller and the Pilot online monitoring system. Section IV is emphasized on the hardware system of the application. Section V shows

low chart of the presented application. Section VI presents the tested results of the application tested results of the application

II. KEIL MDK AND LPC4357 FEATURES

The MDK-ARM is a complete software development environment for Cortex™-M, Cortex-R4, ARM7™ and ARM9™ processor-based devices. MDK-ARM is specifically designed for microcontroller applications. microcontroller devices offer a wide range of communication interfaces to meet any embedded design requirement. However, implementing these interfaces presents software developers with real challenges. Middleware components are essential for developers to make efficient use of the device capabilities.

MDK-Professional includes a number of royalty-free, tightly coupled middleware components which enable developers to more easily implement complex communication interfaces in their applications. Middleware components include[4]:

- Graphical User Interface
- USB Host and Device
- TCP Networking Suite
- Flash File System
- CAN Driver

All middleware components are specifically designed and optimized for ARM processor-based MCU devices. The libraries are seamlessly integrated with the µVision environment and offer a modular design with well documented APIs. time features including fast multitasking, hardware interrupts, both priority-preemptive and round-robin scheduling[6].

Fig. 1 illustrates the Keil Architecture and Development Platform. The basic components of Keil e include Graphical User Interface (GUI).

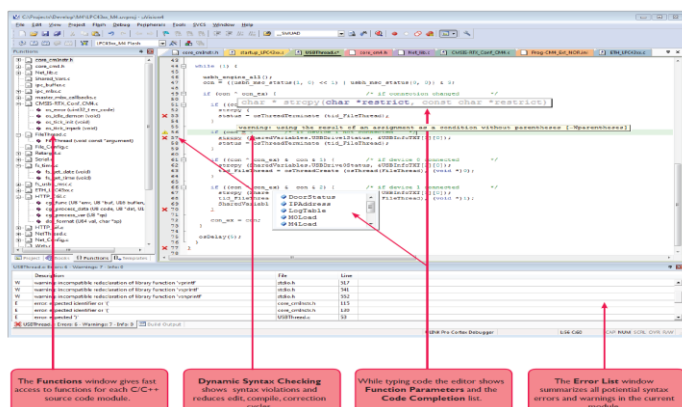


Figure 1. Keil Architecture and Application Development Platform

The GUI Library is a fully featured graphics suite that makes it possible to add graphical user interfaces to embedded applications. It supports a large number of displays and includes tools for rapid GUI creation [2].

- Supports monochrome, grayscale and color LCDs
- Drivers for many displays and display controllers included
 - Window Manager for handling multiple windows
 - Many widget-like buttons, checkboxes and icons available
 - Skinning support for a custom look and feel

- Optimized for speed and size
- Wide range of examples for evaluation boards.

A. Flash File System

The Flash File System allows your embedded applications to create, save, read, and modify files in a wide range of standard storage devices. The Flash File System offers:

- Standard ANSI C File I/O application interface
- NOR and NAND Flash support
- RAM, ROM, and SD/MMC/SDHC Memory Cards
- FAT12, FAT16, and FAT32 formats
- SD/MMC card file-caching
- Reentrant and thread-safe operation
- Simultaneous access to multiple storage devices

B. LPC4357

The LPC4357 development board is a full function evaluation platform base NXP LPC4357 ARM Cortex-M4 processor working at 204MHZ; The LPC4357 board integrate all core components, memory, interfaces and tested software source code, which enable users make a reliable product prototype base it. The board help users to make quick software application and evaluation for NXP LPC4357 ARM Cortex-M4/M0 processor[5]. The LPC4357 board deliver a rich set of memory resource and periphery interfaces, include 32MB SDRAM, 2MB NorFlash and 4MB SPI FLASH, Hi-speed USB Host, audio input/output, 10/100Mbps Ethernet, CAN, JTAG,4.3"/7.0" TFT LCD Module, UART, MicroSD and user GPIO pins. These peripheries interfaces and user extension connector working with full tested source code to build a develop platform helps users evaluate, learning and make prototype base NXP LPC4357ARM Cortex-M4 processor at a fast period[5].

1) LPC4357 development board specifications[3]

- PCB Size: 115mm x 90mm
- PCB layers: 4
- Power supply: 5V/2A
- Debug interface: 20-pin,2.54mm JTAG connector
- Working temperature: -40~+85□
- ROHS Compliance

2) Hardware features Processor[3]

- NXP LPC4357, Cortex-M4, Cortex-M0 dual-core , working at 204MHZ
- 204 MHz, 32-bit ARM Cortex-M4
- 204 MHz, 32-bit ARM Cortex-M0 asymmetrical coprocessor
- 200 kB SRAM for code and data use
- 64 kB ROM containing boot code and on-chip

3) Software drivers

- 32/128-bit One-Time Programmable (OTP) memory for general-purpose use.

4) External memory

- 32MB SDRAM

- 2MB NorFlash
- 4MB SPI FLASH
- 5) *Audio interface*
 - Audio 3.5mm input
 - Dual track audio 3.5mm output
- 6) *TFT LCD*
 - 24 Bit color
 - 80x272 pixel support 1024 x 768
- 7) *Data interface*
 - 3 x UART (UART0, UART2 and UART3, UART2 need external MAX3232)
 - 1x Hi-speed USB HOST
 - 1x Mini USB OTG
 - 1x Ethernet interface
 - 2x CAN interface
 - 1x RS485(reuse with UART1)
- 8) *LED*
 - 1x Power indicator
 - 6x User LED

C. LabVIEW

Lab VIEW is a development environment based on graphical programming. It includes terminology, icons and ideas recognizable to technicians, scientists and engineers. Lab VIEW relies on graphical symbols rather than text-based language to describe programming actions. It is integrated fully for communication with hardware such as GPIB, VXI, CAN, TCP/IP networking, ActiveX and plug-in data acquisition boards [5].

D. Serial Communication

Asynchronous serial communication technology plays an important role in automatic testing field for embedded system, because of its flexibility, facility and reliability. CAN interface is for the transferring information between the devices and debugging of application. Serial communication requires mainly four parameters: the baud rate of the transmission, the number of data bits encoding a character, the sense of the optional parity bit and the number of stop bits [2][3].

E. Control Area Network (CAN)

Development of the CAN-bus started originally in 1983 at Robert Bosch. The protocol was officially released in 1986 at the Society of Automotive Engineers (SAE) congress in Detroit, Michigan. The first CAN controller chips, produced by Intel and Philips, came on the market in 1987. Bosch published the CAN 2.0 specification in 1991 [2].

CAN is a multi-master broadcast serial bus half duplex wired standard for connecting Electronic Control Units (ECUs). Each node is able to send and receive messages, but not simultaneously. A message consists primarily of an identifier, which represents the priority of the message, and up

to eight data bytes. It is transmitted serially onto the bus. This signal pattern is encoded in non-return-to-zero and is sensed by all nodes [3].

The devices that are connected by a CAN network are typically sensors, actuators, and other control devices. These devices are not connected directly to the bus, but through a host processor and a CAN controller. If the bus is free, any node may begin to transmit. If two or more nodes begin sending messages at the same time, the message with the more dominant will overwrite other nodes' less dominant id's, so that eventually only the dominant message remains and is received by all nodes[2][3]. This mechanism is referred to as priority based bus arbitration. Messages with numerically smaller values of ids have higher priority and are transmitted first.

Each node requires a:

- 1) *Host processor*
 - The host processor decides what received messages mean and which messages it wants to transmit itself.
 - Sensors, actuators and control devices can be connected to the host processor [8].
- 2) *CAN controller (hardware with a synchronous clock)*
 - Receiving: the CAN controller stores received bits serially from the bus until an entire message is available, which can then be fetched by the host processor (usually after the CAN controller has triggered an interrupt).
 - Sending: the host processor stores its transmit messages to a CAN controller, which transmits the bits serially onto the bus.
- 3) *Transceiver*
 - Receiving: it adapts signal levels from the bus to levels that the CAN controller expects and has protective circuitry that protects the CAN controller.
 - Transmitting: it converts the transmit-bit signal received from the CAN controller into a signal that is sent onto the bus.

Bit rates up to 1 Mbps are possible at network lengths below 40 m. decreasing the bit rate allows longer network distances. The CAN data link layer protocol is standardized in ISO 11898-1. This standard describes mainly the data link layer composed of the logical link control sub layer and the media access control and some aspects of the physical layer of the OSI reference model. All the other protocol layers are the network designer's choice[2][3].

III. AERO GAS TURBINE ENGINE CONTROLLER AND PILOT ONLINE MONITORING DISPLAY

Aero gas turbine engine controller performs engine control functions as per the control laws apart from providing the monitoring display system with details of various engine parameters. Since engine needs to be monitored and controlled periodically, the presence of engine controller is inevitable. Aero gas turbine engine controller is a critical component of the aircraft. This criticality demands a high reliability of the engine and its control unit. The Aero gas turbine engine controller is controlled by a full authority digital engine control system consisting of hydro mechanical components, actuators,

sensors and digital electronic control unit (DECU). DECU is a rugged and robust industrial computer, which acts as a black box, and runs infinitely and gets data periodically for every 30 seconds from aero gas turbine engine.

The online monitoring system functionality is to monitor, store and analyze engine parameters. This is done through serial communication. As engine controller is critical, expensive and sophisticated component, so the online monitoring system may not be always available for testing. Hence, a simulator is required for testing of online monitoring system in its absence.

The various hardware interfaces of the online monitoring system and their interaction with one another are illustrated in "Fig II". Aero engine comprises of ADC, DAC, CAN interface, timer, Pentium III processor, power cards etc which is used in embedded controller display. There is a requisite for real time communication between aero gas turbine engine controller and pilot online monitoring display. This is accomplished by using CAN serial communication. The application software which is designed and developed will be real time and it takes care of serial communication protocol.

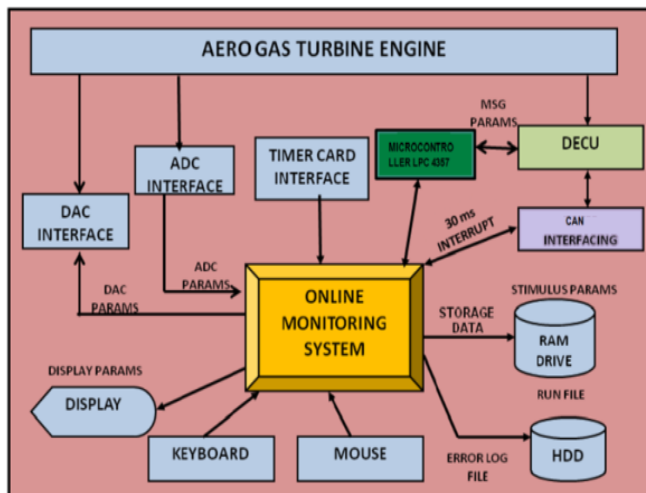


Figure 2. Hardware Interface Diagram of the Monitoring System

IV. HARDWARE SYSTEM

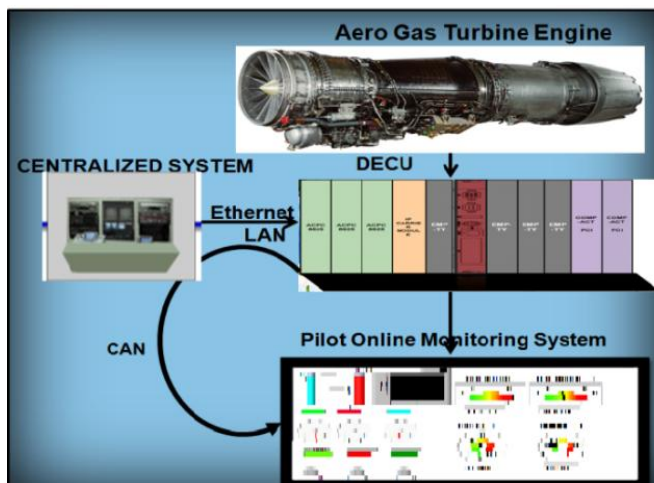


Figure 3. Block Diagram of hardware system

The Fig. 3 shows block diagram of cross development hardware system. Hardware system comprises of following.

Host PC: Host PC consists of host operating system i.e. windows 2000. This host operating system has a target operating system i.e. Keil MDK Pro can be configured and scaled based on the application.

A. Ethernet LAN

Ethernet LAN acts as a media for communication between host PC and DECU.

B. Engine controller

Engine controller is Commercial Off The Shelf component (COTS) which controls the engine.

C. CAN

CAN acts as a serial communicator between engine controller and pilot critical display.

D. Pilot critical display

Pilot critical display is a standalone application which gets the information from engine controller periodically for every 30milisecond which is needed to be displayed. This is developed by using Lab VIEW.

The hierarchy of the design carried is as follows.

Initially, the configuration of CAN serial communication port COM-1 is done. The data taken from aero gas turbine engine to the Digital Electronic Control Unit (DECU) is processed for 100 milliseconds. The data residing in the DECU is converted into packets for the communication between DECU and pilot online monitoring system. Then the packets are transferred to the pilot online monitoring display system through the configured CAN serial communication port COM-1.

V. FLOW CHART

The flow chart of the presented cross developed application is as shown in "Fig IV". It includes four stages.

- In stage1, the Microcontroller LPC 4357 senses the various parameters of the Engine at different points of engine run .
- The data format of CAN is as shown in "Fig VI" which includes SOF (Start Of Frame)bit, Arbitration ID,RTR(Remote Transmit Request),IDE (Identifier Extension),DLC(Data Length Code),Data Bytes, CRC(Cyclic Redundancy Check) .
- Every 30milisecond updated data is being packed for transmission.
- The processed data is transmitted to online monitoring system to the display for the pilot on real time.

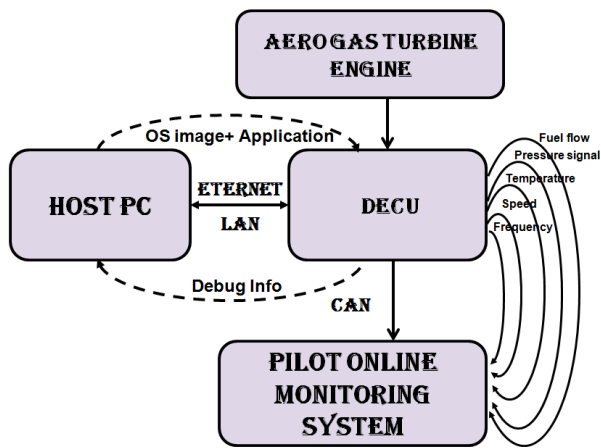


Figure 4. Flow Diagram of CAN Serial Communication

Fig. 5 describes The Fault Handling Techniques and the operation process [7]:

1. Assume that the system is running with copy 0 as active unit and copy 1 as standby.
2. When the copy 0 fails ,copy 1 will detect the fault by any of the fault detection mechanisms.
3. At this point, copy 1 takes over from over from copy 0 and becomes active. The state of copy 0 is marked suspect, pending diagnostics.
4. The system raises an alarm ,notifying the operator that the system is working in a non-redundant configuration.
5. Diagnostics are scheduled on copy 0 .This includes power on diagnostics and hardware interface diagnostics .
6. If the diagnostics on copy 0 pass ,copy 0 is brought in service as standby unit. If the diagnostics fail, copy 0 is marked failed and the operator is notified about the field card.
7. The operator replaces the failed card and commands the system to bring the card in service.
8. The system schedules diagnostics on the new card to ascertain the card is healthy .
9. Once the diagnostics pass, copy 0 is marked standby.
10. The copy 0 now starts monitoring the health of copy 1 which is currently the active copy .
11. The system clears the non-redundant configuration alarm as redundancy has been restored .
12. The operator can restore the original configuration by switching over the two copies.

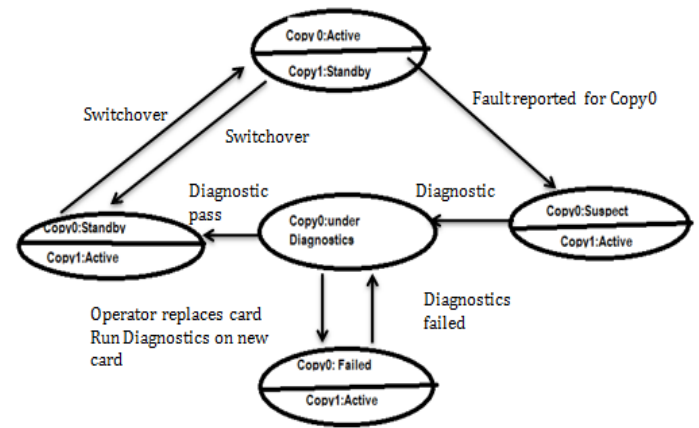


Figure 5. Fault Handling Life Cycle

S	11-BIT	S	18-BIT	R	DLC	0...8	CRC	A
O	ARBITRATION ID	I	ARBITRATION ID	T		BYTES DATA		C
F		D		R				O
		E						F

Figure 6. Data Format

VI. TEST RESULTS

Due to non availability of actual system, several systems can be tested with simulator. The proposed application is tested for the performance in terms of serial communication interface and exchange of expected messages using pilot online monitoring system simulator. The test results are as shown in Fig. 7 and Fig. 8. The test results show that the proposed application is developed on the target board and information is transmitted to pilot online monitoring system. The application is successfully developed using serial communication interface and message is transmitted between aero gas turbine engine controller and pilot control display system.

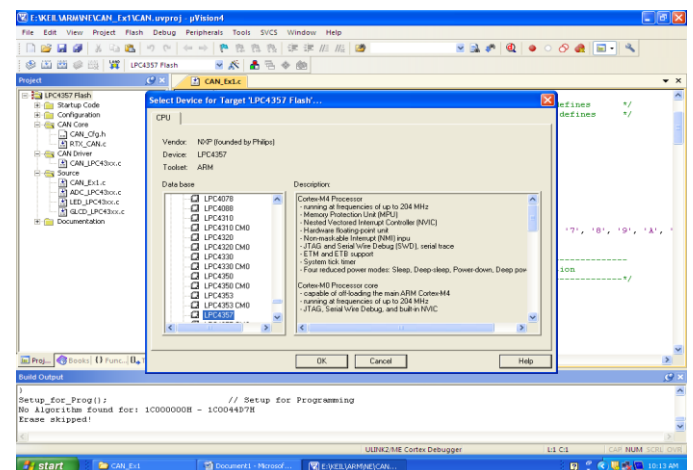


Figure 7. Test Result using Keil MDK

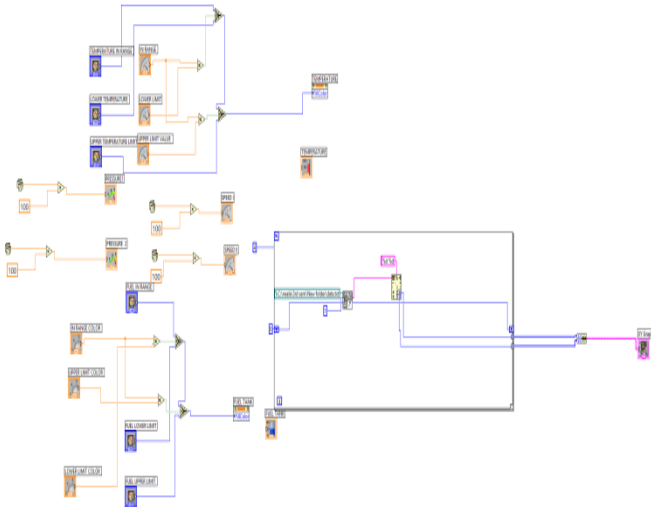


Figure 8. Test Results using Block diagram of LAB View

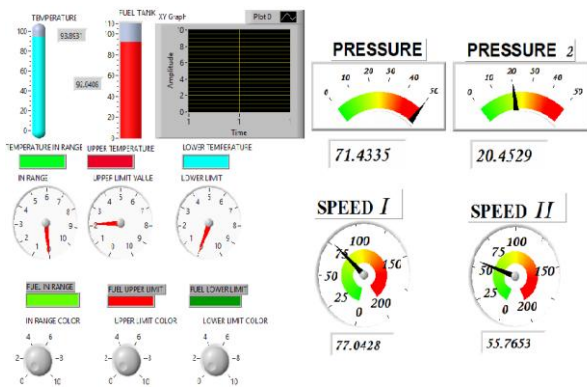


Figure 9. Test Results using Front Panel of LAB View

VII. CONCLUSION

The design and development for real time communication between aero gas turbine engine controller and pilot online monitoring display is successful. Aero gas turbine engine controller is enclosed by an unique polymorphic approach which could adapt itself easily to any testing environment videlicet rig, ground, altitude, temperature, fuel intake, power dissipated, pressure, amount of air taken, speed etc. To ensure deterministic performance, the application is developed using CAN AND for communication based and fault tolerance on the Keil MDK platform and implemented through Lab VIEW execution system. Tested for data fault tolerance and redundant system.

REFERENCES

- [1] Prindle.j, "A method for providing turnkey embedded computer operating systems for independently developed user applications in a military environment", digital avionics system, 18th IEEE conference, vol-1, 1999, pp .A.5-2.A.5-8.
- [2] Shuangyou Wang, Zhi'an Wang and Xuhui Wang "Multichannel temperature acquisition System based on ARM and CAN", International conference on computer, control engineering, pp 33-34, 2010.
- [3] Zhu qishen , Zhu dongmei and Su xunhon "Distributed remote temperature monitoring and acquisition system based on CAN bus", 2010 prognostic and system health management Conference, 2010.
- [4] Getting started creating application with μ vision 4
- [5] ARM® Keil® MDK™ toolkit featuring Serial Wire Viewer and ETM Trace For the Keil MCB4357™ EVAL board Version 1.0 Robert Boys
- [6] "Sam Seiuort",Real Time Operating Systems
- [7] Rajendra Prasad, "Fault Tolerance in Aerospace Systems", Published and Printed by D Murugan,OM offset ,sharada industrial estate, Sinhagad road ,Pune -51,Maharashtra.
- [8] AWAECMA Project Report on PSM+ICR, <http://www.faa.gov/certification/aircraft/engine-sm+cr.doc> , 1998.
- [9] Christine M. Belcastro, Celeste M. Belcastro, "Application of failure detection, identification, and accommodation methods for improved aircraft safety", *Proceedings of the American Control Conference*, Vol. 4, 25- 27,pp. 2623 -2624, June 2001.